Database Assignment

# Name:

## Syed Muhammad Mubashir Rizvi

# Roll Number:

## CS-20071

# Department:

## Computer and Information Systems Engineering

# Submitted to:

## Miss Maria

# Date:

## 28th March 2021

# *Basic Employee Database:*

An application that works on simple records of employee and manages their data. The application allows the user to save or load from file, input new records, delete previous records either from the file or the new records the user entered. It is a GUI based application. The application keeps a track of the following fields:

## Fields:

1. Name
2. Age
3. Date of Birth
4. Email Address
5. First day of employment
6. Contact
7. ID (must be of 7 characters)
8. Wage per day
9. Total working days
10. Retirement (60 years limit, and which remains only on display and is not saved in file
11. Monthly Wage (a calculated field that is shown on the display, it is not saved in file, it is calculated using “wage per day” and “total working days”)

## Modules and Widgets Used:

1. Tkinter
2. Label
3. Button
4. Entry
5. LabelFrame
6. Messagebox :

pops up a dialog box, can show some message, ask some question, ask for confirmation, or show some error also.

1. Filedialog :

this widget is used to work with files, by using this widget, one can open or save files or directories in the same manner when one saves a file in a computer system.

1. Simpledialog :

it shows a pop up dialog box, and it asks some input and that information can be saved in a variable and used later in the code. I have only used askinteger and askstring in my code but there are many other options other than these two.

1. Listbox :

it is useful to for a number of things, such as, it allows to display a large data, and the user can also manipulate that data according to his/her choice by creating some specific buttons. In my code, I have only used this to display the records.

1. Scrollbar :

as the name indicates, it adds a scroll bar, for that I made a frame, in which I added the “listbox” and the “scrollbar”. The scrollbar widget allows to add both horizontal and vertical scrollbars, in my code I used this widget to add a scrollbar vertically so that the user can manipulate the data if it exceeds the normal limit of a window.

1. Time Module :

t.sleep = it sleeps or stops at the current position for the number of seconds passed to it.

t.localtime = it returns the current time, date, day, hour, seconds etc.

t.time = it is called a timestamp, which shows the number of seconds that has passed since the “epooc”, which represents the point where time starts. Python uses Unix Epooc, i.e 1st January 1970 at 0hr, 0min, 0sec.

Implementation:

I used t.time() to get the number of seconds, then in it I added the number of seconds of the age entered by user (the age was subtracted by 60), for e.g, is user entered his age 20, then the program would do 60 – 20 and would add the seconds of 20 years into the timestamp “t.time”. I then passed that variable to “t.localtime()” which converts those seconds into a format of year, month, day, hour, min, sec etc. I then just fetched the year from this variable and predicted the retirement age of the user.

## Functions:

### delt\_entry( ):

This function simply deletes/clear the text in an entry box.

### data\_save\_Shell( ):

This function fetches data user entered in the entry boxes, checks the requirement of age (i.e less than 60) and the requirement of ID (i.e must be of 7 characters). The function then makes a list of all the data and saves that list as a single record and that record is then saved into a variable “entered\_records” present in the main code.

### add\_new( ):

This function asks the user for confirmation to save previous data, after that the function “data\_save\_Shell()” is called. The function then asks for confirmation to remove precious data, and after getting confirmation “delt\_entry” is called to clear the entries, and it also removes the data present if user checks the retirement.

### load\_file( ):

This function first opens a dialog box which allows the user to select a file of his/her choice. After file selection, the data is read, and then saved inside a variable “loaded\_records” present in the main code. After loading, a message is printed saying that it was a successful load.

### display\_records( ):

This function first accesses the variables “entered\_records” and “loaded\_records” and also accesses a global variable “list\_of\_all\_records” which makes a new list of both the combined records. This function then created a new window with a “listbox and a scrollbar” for displaying data in an easy way. The new list (list\_of\_all\_records) is then iterated and the data is displayed on this window by inserting it in the “listbox”.

### save\_file( ):

This function also accesses the modified “list\_of\_all\_records” from the main code and then asks the user to select a file for saving the data in. This function basically saves those records that were displayed to the user. The variable is then iterated using a loop and the records are written in the selected file. One record per line. In the end, the function shows a message saying that the records were saved.

### delete\_record( ):

The function accesses the “listbox” and “list\_of\_all\_records” and asks the user if he wants to delete a record from current display of records or delete a record from a file. In both the cases, the user is advised to enter the ID of the employee, that record is then searched, and deleted, the record is also removed from the “list\_of\_all\_records” in the first case, the new display is also shown to the user if he/she had selected to delete some record that was displayed. While in the second case, the file is selected by the user and it is read, searched for the record, it removes it from the content, the file is then opened again in write mode, and the new content is written.

### exit\_app( ):

This function asks the user if the data that was displayed was saved in any file, the user is then asked if he wants to save it in some file or no. If the user has saved it, he/she is asked for confirmation to exit the app. If not, he/she is asked the choice to save the data in a file.

### calculate\_retirement\_button( ):

This function fetches the age from the entry box of age and checks if it is less than 60. The module “time” is used to calculate the year in which the user will retire, after that the calculated information is displayed on the main interface.

## Working:

The main interface of the application shows the entities to be filled by the user, it also shows the functions or buttons used in the program. The user can then keep on adding records and saving them in a python variable for time being by using the add\_new() function. The user can also load some file of his choice and the data of that file is also stored in a different python variable for the time being. The user can display all of the information (the entered records and the loaded records) in a different window, s/he can then select if s/he wants to delete some record which is displayed on the new window, for that to happen, the ID of the record to be deleted is required. The same delete function can also be used to delete some record in a file and for that to happen the user is requested to select a file from which s/he wants the record to be deleted. Finally, the user can exit the app.

## Drawbacks:

* All of the entities are not checked for valid data, only the age and ID fields are checked.
* Loading file, here the program is somewhat restricted in such a case if the user opens such a file which was not saved using this application, because in this application the data in the file is written in such a way that there is 1 record per line and the data of each record is separated by underscore (“\_”).
* When asking for ID, there is not data validation whatsoever so the application is dependent on user to enter the ID correctly, neither there is a pop up message showing some error if the entered ID does not match any ID in the records.
* The user is advised to first display all the records before deleting them from the display menu, so that it is easier for the user to see the changes in the display accordingly, because the user would have to switch between windows.

## References:

* <https://www.python.org/>
* <https://www.geeksforgeeks.org/>
* <https://stackoverflow.com/>